nginx configuration

Cache

- service reload : nginx 관련 conf 파일을 서버에 반영해주는 작업 (reloading config file), restart는 권장되지 않는다고 하는데 이유는 무엇인지?
- error_log > crit : 프로세스에 영향을 줄 수 있는 오류
- nginx pid : /var/run/nginx.pid 에서 확인 가능하며 download 서버는 20045
- cache는 요청하는 URL에 따라서 좌우된다. 예를 들어 x1Util.js 파일을 다운로드 하고싶을 때 페이지에서 **같은 이름의 js 파일을 요청 시 만약 해당 파일에 대한 캐시가 저장되어 있다면 캐시에 저장된 파일**을 가져온다. 하지만 우리는 새로운 파일을 반영해야한다. 따라서 변경할 때에는 preset에서 호출하는 **x1Util.js 뒤에 특정문자(version)을 추가하여 캐시를 참조하기 않도록 해주는 것**이다.
- https://jojoldu.tistory.com/60

  https://yongdev91.tistory.com/1

→ nginx 모듈은 configuration 파일 내의 **directives** 에 의해 제어된다. 종류는 simple directive, block directive 로 나눠진다.

- simple directive
  1. user : nginx 서버의 동작시킬 시스템 사용자의 ㅇ름
  2. worker_processes : 몇 개의 thread가 사용될지, 보통 서버의 core 수에 맞추어 작성하는 것이 권장되며 그 이상도 가능하다. 또한 auto 기능을 통해 자동 설정 가능.
  3. pid : nginx pid가 작성되어있음. /var/run/nginx.pid 에서 확인 가능
  4. include : 외부 configuration 파일의 내용을 가져옴.

- block directive

  내부에 (properties로 이해하면 될 것 같은) directives이 추가적으로 다시 들어가는 경우를 context라고 한다. context에는 대표적으로 events, http, server, location directive들이 있다. 어떤 Context에도 속하지 않고, 바깥에 나와있는 directive 들은 **main context** 에 속해있는 것으로 간주한다.

→ 설정은 보통 context에서 이루어지며 주로 server, http, event 등이 있다.

- HTTP Context
  1. include : 참조할 configuration 내용을 참조
  2. access_log
  3. autoindex: 디렉토리 내의 파일들을 목록화하여 파일로 다운로드하는 기능
  4. send_file : nginx 에서 정적파일을 보낼 수 있도록 설정
  5. tcp_nopush : 클라이언트로 패킷이 전송되기 전 버퍼가 가득찼는지 확인하고 다 찾을 때에 패킷을 젓농하도록 하는 기능. 네트워크 오버헤드를 줄이기 위해 설정한다.
  6. keepalive_timeout : 클라이언트와 서버간의 keepalive 커넥션의 유지시간을 설정, 너무 긴 경우 커넥션 연결에 따른 자원 낭비가 발생할수도 있기 때문에 적절한 시간을 설정해야함

7. gzip : 컨텐츠 압축을 진행한다. 가상 호스트(server) 마다 진행하지 말고 메인 설정 파일 /etc/nginx/nginx.conf 의 HTTP 블록에 설정하면 전역으로 작동하여 편리함
    - gzip (on) : 컨텐츠 압축 사용을 킴
    - gzip-proxied (any) : 항상 압축을 진행함.

      → off (proxy 요청 시 압축 X), expired(요청 헤더에 expires가 존재+만료돈 경우에만 압축), no-cache(요청 헤더에 Cache-control 존재 + no-cache인 경우에만 압축), no-store(요청 헤더에 Cache-control 존재 + no-store인 경우에만 압축)
    - gzip_types : 컨텐츠의 유형에 따라 해당 옵션 지시자를 통해 압축 여부를 설정할 수 있다.
    - gzip_vary : gzip, gzip_static, or gunzip의 설정들이 on으로 되어있을 때 응답 헤더에 “Vary: Accept-Encoding”를 넣을지 말지에 대한 명령어
    - gzip_disalbe : 요청 헤더의 User-Agent와 입력한 정규식이 일치하면 압축 X

    > 💡 *gzip directive에서 comp_level(압축 정도), min_length(압축 최소 사이즈), buffers (버퍼의 숫자, 크기) 부분을 설정해본적이 있는데 효과가 없었는지?*

8. proxy_cache_path
    - path : 캐시 내용이 로컬 어디에 저장될 것인지

      /etc/nginx/cache
    - levels : directory depth와 사용할 name의 길이

      1:2 → /data/nginx/cache/c/29/b7f54b2df7773722d382f4809d65029c

        level을 설정하지 않는 경우는 현재 디렉토리가 되고 1: 은 이후 첫번째 단계의 디렉토리의 글자, :2 는 두번째 단계 디렉토리의 글자 수

        → c/29
    - keys_zone : 캐시 키로 사용할 이름:크기

      img_cache:10m
    - max_szie : 캐시 파일의 maximum 크기, 초과시 오래된 순으로
    - inactive : access 되지 않는 경우 얼마나 뒤에 삭제할 것인지
    - use_temp_path : 설정한 path 외에 임시 저장 폴더를 사용할건지

9. location : 요청 URL 캐치할 패턴 작성

    → location ~* \. (?: css | js | gif | png | jpg | peg)
    - ~* : 대소문자 구분 X, 정규표현식과 일치
    - \ : 여러 페턴 요소를 한 단위로 묶으며?
    - ?: 없거나 1회?

10. proxy_cache : 캐싱에 사용할 key-zone(위에서 설정한), prxy_cache_valid 는 캐싱할 response_code와 시간, response header의 Cache-contorl이 우선순위가 높다

11. proxy_cache_key : 캐시 키를 어떻게 저장할 것인지 스키마를 지정

    →$scheme$request_method$host$uriis_argsargs

    http/GET/img.x1.co.kr/x1/expert_img/519/img_onair519.png/ /

12. proxy_cache_valid : 200 302 20m; 200, 302에 대해서는 20분간 유지
13. expires : 캐시 만료 기간, 1days
14. add_header
    - X-Proxy-Cache $upstream_cache_status :
      X-Proxy-Cache 헤더에 HIT, MISS, BYPASS와 같은 캐시 적중 상태정보가 설정
    - Cache-Control "public" : 응답이 어떤 캐시에 의해서든 캐시된다는 것을 나타냅니다.

      → location에 해당 설정이 들어가게 되면 클라이언트의 Cache-Control 헤더 요청을 무시한다. 또한 WAS에서 응답하지 않고 캐시 서버가 응답한다. nginx
15. ssl_session_cached shared:SSL:10m; 연결 자격증명에 대한 Cache량을 늘려 Session 매개변수의 재사용률을 높여준다. (이미지 캐시랑은 관련 없는듯)

→ **Q**

4/c8/~ : _=1619651471815
e/5e/~ : _=1619651416592
4/19/~ : _=1619670444038

이렇게 총 열몇개가 있는데 원래 캐시라면 하나의 파일은 jquery min.js 파일은 변하지 않으니 캐시가 하나만 있어야 하는게 아닌지?

proxy_cahce_key 스키마와 관련된 건지,,? is_args,args

Host

- client_max_body_size : HTTP Client 의 Content-length 헤더값이 관련, 브라우저에서 파일을 업로드 시 체크하는 최대 용량
- proxy_connect_timeout : 프록시 서버에 연결을 수립 시에 최대 시간, 보통 75초를 초과하지 않는다? → 300?
- proxy_send_timeout : request 요청을 proxy server로 송신하는 경우 최대 타임아웃, 연결되지 않는 경우 connection은 종료된다.
- proxy_read_timeout : response 응답을 읽는데에 소요되는 최대 타임아웃
- send_timeout : 클라이언트에게 response을 송신하는데 소요되는 최대 타임아웃
- set_real_ip_from : reverse_proxy 로 설정된 서버의 ip를 작성

Server

- proxy_pass : 세팅한 upstream, 어떤 요청을 받을 것인지
- proxy_set_header : 프록시 서버로 전달되는 header의 종류마다 설정해줌

  → nginx가 proxy 서버에 요청할 때 default로 정의되는 헤더는 Host, Connection 두 가지이다.

nginx.service

https://github.com/systemd/systemd/issues/16184

https://man7.org/linux/man-pages/man8/systemd-socket-proxyd.8.html

https://www.joinc.co.kr/w/man/12/nginx/static

https://m.blog.naver.com/PostView.nhn?blogId=malloc813&logNo=220511432388&proxyReferer=https:%2F%2Fwww.google.com%2F

https://access.redhat.com/discussions/3027351

https://www.nginx.com/blog/nginx-caching-guide/

1. privateTmp=True는 systemd가 구동시킨 서비스에 /tmp 디렉토리 하위에 systemd-private-bf2c4997399145c18ace3dc1313dde81-nginx.service-Kgu2pX 를 생성시킨다. 그리고 해당 디렉토리를 /tmp로 인식하도록 마운트 시킨다.

   https://m.blog.naver.com/PostView.nhn?blogId=malloc813&logNo=220511432388&proxyReferer=https:%2F%2Fwww.google.com%2F
2. nginx 최초 캐싱이 될 때에 캐시를 위한 임시 저장소 영역에 파일을 작성한다. 이후 use_temp_path=off 지시자가 nginx가 캐시된 곳과 같은 디렉토리에 쓰도록 지시한다.

개발 서버 테스팅 결과

→ x1Util.js / common.css 파일을 변경 후 버전업한 뒤 개발서버에서 테스팅을 진행했다.

1. x1Util

   → 버전업을 진행했음에도 불구하고, etag 및 last-modified가 변경되지 않았다.

2. common.css

   → 버전업을 진행한 뒤 에는 etag, last_modified가 모두 변경되면서 cache가 MISS 됨.

Q.

- 두 파일의 변경 위치 및 내용은 동일했으며 Cache-Control 설정도 동일했다. 하지만 한 파일에서만 변경 작업이 이루어졌다.
- 프록시 서버에서는 last-modified 관련 설정을 진행한 적이 없는데 자동으로 이루어지고 있다.

  → 여기서 우리가 변경하는 설정 public 은 response header에 붙어나온다

- **no-cache** vs **Must-Revalidate**

신선하지 않은 사본을 원 서버와의 최초의 재검사 없이는 제공해서는 안 된다는 것을 의미한다.

- no-cache와 must-revalidate는 재검사를 거쳐 응답을 하는 공통점이 있지만, 재검사 주기가 차이점이 된다.

  no-cache는 매번 재검사를 하고, must-revalidate는 최초에 재검사를 한다.

  응답을 10초 동안 캐시할 수 있다면, 10초 후에 다시 재검사. no-cache 0초라고 생각하면 된다.

  파일의 양에 따른 변화는 큰 차이가 없다. (1 vs 20 line)

privateTmp 파일 수정 시 systemctl reload application.service

http://oniondev.egloos.com/9972854

https://stackframe.tistory.com/39

개발 서버에서는 nginx.service 의 원본 파일이 /lib/systemd/system에 있다.